Assignment 2: Coding Basics

Aubrey Knier

OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics on coding basics.

Directions

1. Change "Student Name" on line 3 (above) with your name.
2. Work through the steps, **creating code and output** that fulfill each instruction.
3. Be sure to **answer the questions** in this assignment document.
4. When you have completed the assignment, **Knit** the text and code into a single PDF file.
5. After Knitting, submit the completed exercise (PDF file) to the dropbox in Sakai. Add your first and last name into the file name (e.g., "FirstLast_A02_CodingBasics.Rmd") prior to submission.

Basics Day 1

1. Generate a sequence of numbers from one to 100, increasing by fours. Assign this sequence a name.
2. Compute the mean and median of this sequence.
3. Ask R to determine whether the mean is greater than the median.
4. Insert comments in your code to describe what you are doing.

```
#1. creating a sequence from 1 to 100 by 4 (seq(from,to,by)), and assigning the name "hundred_seq" to t
hundred_seq <- seq(1,100,4); hundred_seq
```

```
##  [1]  1  5  9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
```

```
#2. using mean() function to calculate mean of hundred_seq vector and naming the output
mean_hundred_seq <- mean(hundred_seq); mean_hundred_seq
```

```
## [1] 49
```

```
#using median() function to calculate median of hundred_seq vector and naming the output
median_hundred_seq <- median(hundred_seq); median_hundred_seq
```

```
## [1] 49
```

```
#3. using the conditional statement ">" and the named mean and median outputs to determine if TRUE or F
mean_hundred_seq>median_hundred_seq
```

```
## [1] FALSE
```

Basics Day 2

5. Create a series of vectors, each with four components, consisting of (a) names of students, (b) test scores out of a total 100 points, and (c) whether or not they have passed the test (TRUE or FALSE) with a passing grade of 50.

6. Label each vector with a comment on what type of vector it is.

7. Combine each of the vectors into a data frame. Assign the data frame an informative name.

8. Label the columns of your data frame with informative titles.

```
names <- c("Lucy", "Ben", "Jack", "Olivia") #character vector
scores <- c(82,91,43,70) #numeric vector

PF <- function(x) {
    grade <- x>=50
    return(grade)
  }

pass.grade <- PF(scores)
pass.grade #logical vector
```

```
## [1]  TRUE  TRUE FALSE  TRUE
```

```
gradebook <- cbind(names,scores,pass.grade)
colnames(gradebook) <- c("Student Name", "Test Score","Passed?")
gradebook
```

```
##      Student Name Test Score Passed?
## [1,] "Lucy"       "82"       "TRUE"
## [2,] "Ben"        "91"       "TRUE"
## [3,] "Jack"       "43"       "FALSE"
## [4,] "Olivia"     "70"       "TRUE"
```

9. QUESTION: How is this data frame different from a matrix?

   Answer: This data frame has multiple different modes (character, numeric, and logic), whereas in a matrix, all columns would have to have the same mode.

10. Create a function with an if/else statement. Your function should determine whether a test score is a passing grade of 50 or above (TRUE or FALSE). You will need to choose either the `if` and `else` statements or the `ifelse` statement. Hint: Use `print`, not `return`. The name of your function should be informative.

11. Apply your function to the vector with test scores that you created in number 5.

```
if_else_PF <- function(x) {
  if(x>=50) {
    print("Passed")
  }
  else{
    print("Failed")
  }
}

if_else_PF(scores) #doesn't work with vectors
```

```
## Warning in if (x >= 50) {: the condition has length > 1 and only the first
## element will be used

## [1] "Passed"
```

```
#have to do each student individually
Lucy_score <- if_else_PF(82); Lucy_score
```

```
## [1] "Passed"

## [1] "Passed"
```

```
Ben_score <- if_else_PF(91); Ben_score
```

```
## [1] "Passed"

## [1] "Passed"
```

```
Jack_score <- if_else_PF(43); Jack_score
```

```
## [1] "Failed"

## [1] "Failed"
```

```
Olivia_score <- if_else_PF(70); Olivia_score
```

```
## [1] "Passed"

## [1] "Passed"
```

```
ifelse_PF <- recipe6 <- function(x){
  ifelse(x>=50, "Passed", "Failed") #log_exp, if TRUE, if FALSE

}

ifelse_PF(scores) #works with vector, and returns output vector the same length
```

```
## [1] "Passed" "Passed" "Failed" "Passed"
```

12. QUESTION: Which option of `if` and `else` vs. `ifelse` worked? Why?

    Answer: The "ifelse" function worked because the input can be a vector, and the result will be a vector the same length. When I first tried the "if" and "else" option, I could not simply input the scores vector I created and had to individually run each student. When I created an "ifelse" function, I was able to simply input the scores vector and receive a vector output with four reponses corresponding to the student names.